Creating Charts by Demonstration

Brad A. Myers
bam@cs.cmu.edu

Jade Goldstein
jade@cs.cmu.edu

Matthew A. Goldberg
mg4q@andrew.cmu.edu

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

"Gold" is a new interactive editor that allows a user to draw examples of the desired picture for business graphics and the system automatically produces a visualization. To specify a custom visualization in other systems, code must be written or a bewildering array of dialog boxes and commands must be used. In Gold, as the user is drawing an example of the desired visualization, knowledge of properties of the data and the typical graphics in business charts are used to generalize the example and create a picture for the actual data. The goal is to make designing a complex, composite chart almost as easy as sketching a picture on a napkin.

KEYWORDS: Data Visualization, Demonstrational Interfaces, Interactive Techniques, Business Charts.

INTRODUCTION

When users want a chart or graph of their data in personal computer programs like Lotus 1-2-3, Microsoft Excel, or DeltaGraph Professional [4], they select the range of data, and the system will use some simple heuristics to automatically display the data in a chart. Alternatively, the user can select from a menu of pre-defined chart types. However, if the system does not choose correctly, and the built-in charts are not appropriate, then it is quite difficult for users to specify the desired pictures. Typically, many complex dialog boxes and commands must be used. This is a significant recognized problem with all of these programs, which so far has not been solved. In fact, users of Microsoft Excel found this process so difficult that the Windows version provides a so-called "Wizard" interface, that takes the user through a set of question-and-answer dialogs. However, this can be tedious and still does not provide the user with sufficient flexibility to easily specify desired displays. Creating custom displays is also difficult with scientific visualization systems like the IBM Visualization Data Explorer. In these, code must be written, either using conventional or visual programming languages.

CHI94-4/94 Boston, Massachusetts USA
© 1994 ACM 0-89791-650-6/94/0106...$3.50

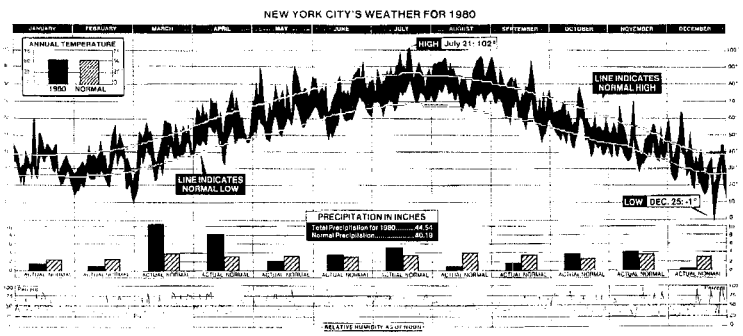
The problem is particularly difficult when there are multiple data values that need to be displayed in the same or linked graphs. Imagine trying to specify a display like figure 1 using the conventional techniques. Also, there is evidence that when users *construct* their own visualizations, they understand the data better than when a visualization is produced for them [19]. Furthermore, a single display is often insufficient, and users need to *explore* the data by changing display types, zooming, rescaling and aggregating data.

To solve these problems, we are building an interactive drawing editor called "Gold" which allows the user to draw parts of an example of the desired display, and the computer will interpret the drawing using knowledge of the usual kinds of displays, and then create a complete drawing based on the actual data. Gold, which stands for Graphs and Output Laid-out by Demonstration, will also let the user change a display using similar techniques. This will make it easier to edit the display and explore the data. Unlike simple drawing tools like MacDraw, the user only needs to draw an example of one or two data elements, which do not have to even be the correct sizes, and Gold automatically will draw the rest. Furthermore, if the data changes, the system automatically updates the picture.

Gold will support almost all the kinds of two-dimensional charts provided by Excel, DeltaGraph [4] and similar programs, including (in the terminology of DeltaGraph): column charts, stacked column charts, bar charts, stacked bar charts, line charts, pie charts, area charts, scatter charts, high-low charts, and range charts. Of course, given the system's demonstrational nature, the user does not need to know what these names refer to, and can instead draw a picture. Furthermore, Gold's real power will come from making it easy to express *combinations* of different charts, and controlling the *parameterization* of properties of charts, for showing relationships among multiple data sets (as in figure 1). This means that with Gold, the user can create charts (such as Myers-Color-Plate-3) that are not even possible with tools such as Excel and DeltaGraph.

RELATED WORK

Data and scientific visualization are very active areas of research with a number of conferences and journals devoted to them. However, it appears that most research is directed at new kinds of visualizations. Our research, however, is aimed at making fairly conventional visualizations much easier to produce.



New York Times, January 11, 1981, p. 32.

Figure 1:
A complex composite chart that summarizes 2,200 numbers. From [20, p. 30]. Our goal is to allow users to easily construct this by drawing only a few elements. (At the time of this writing, however, Gold cannot handle this chart because it does not support linking multiple axes.)

A number of systems have investigated how to automatically produce an appropriate and attractive display given the properties of the data. Examples of this type include APT [7], SAGE [16], BOZ [1], and commercial products like spreadsheets (Microsoft Excel and Lotus 1-2-3) and interactive graphing packages (e.g., DeltaGraph [4]). The commercial products also provide large libraries of built-in displays. Gold makes creating custom displays much easier by allowing the user to sketch the desired display and directly manipulate the results.

SAGEBRUSH [17], which is built on top of SAGE [16], is a new interactive editor that takes an alternative approach to Gold. Rather than having users draw examples of the desired display, in SAGEBRUSH the user assembles displays by selecting graphical objects and assigning data to their properties. In the future, we plan to explore combining the best features of Gold and SAGEBRUSH, and possibly moving Gold to use the underlying SAGE system.

Specifying custom displays is also difficult in scientific visualization systems. "Scientific visualization" is a form of data visualization that presents large amounts of data generated by some scientific observation or simulation. Many of today's scientific visualizations are produced using programming libraries of routines, so the users must write code to specify the desired graphics. Sometimes, a charting routine from a library can be used, but if these are flexible, they generally require specifying lots of complex parameters. Most interactive scientific visualization tools, such as AVS from Stardent and the IBM Visualization Data Explorer, use a dataflow model, where the user graphically wires together nodes that process the data. To specify the particular types of display, the user wires the outputs to a charting icon chosen from a large library. These icons will have many parameters which can be set using dialog boxes or by wiring in the appropriate data. It is not possible to directly manipulate the generated pictures to change the display (other than simple manipulations like rotation and moving a clipping plane).

Gold uses *demonstrational techniques* [13] to generalize from the example drawing to produce the visualization. Many demonstrational systems have been created for other domains, such as user interface construction [9, 14], technical drawing [8], text editing [12], and automating repetitive actions [2]. A new book provides a comprehensive survey [3]. One research system that has begun exploring demonstrational visualization allows the user to draw one example of the layout for objects, and the system generalizes to any number of objects [5]. This system is quite limited, however, and only deals with the layout of rectangles in hierarchies.

The term "data visualization" also applies to systems that create pictures of data from running programs, to help with debugging and understanding [10]. One system that uses demonstrational techniques in an editor to specify algorithm visualizations is Dance [18], which allows users to draw pictures for the graphical elements and attach preprogrammed "path" animations to control the behaviors. However, the pictures for algorithm visualization are quite different from business graphics, so Gold uses entirely different techniques than Dance.

OVERVIEW

The goal of this research is to make specifying a custom visualization to the computer almost as easy as it would be to a tell a professional graphic designer what display you have in mind. Thus, we want to allow the user to quickly draw an example of the desired display, like one might do on a blackboard or napkin. Rather than try to use gesture recognition, however, Gold presents an interface like a conventional drawing program such as MacDraw. By using the conventional drawing operations the user is familiar with, Gold makes the creation of custom charts straightforward. Figure Myers-Color-Plate-1-a (in the back of the proceedings) shows the drawing area in the center, the palette of items that can be drawn on the left, the line and fill style palettes at the bottom, and the pull-down menus at the top. There are three important additions over a MacDraw-like editor. First, the axis primitive at the top of the palette is used for drawing horizontal or vertical axes. Second, clicking on the mark item at the bottom of the palette (figure Myers-Color-Plate-1 shows a star) pops up a menu of marks that can be placed in the window. Then, clicking in the drawing window places a mark of a standard size, but pressing and dragging out a rectangle makes a mark of any desired size. Third, as each graphical element is drawn, Gold creates a "link-box" which is used as feedback for what data is attached to that element. The link-boxes are shown in gray in figure Myers-Color-Plate-1-a.

The data for graphs in Gold is displayed in a spreadsheet-like interface, shown in figure Myers-Color-Plate-1-b. Data can be read into Gold from other spreadsheets, from databases, or from simulations.[1] The user can select a

[1]Generating a correct query to a database to find the data and filtering the data to get it into the correct format can both be difficult tasks for users, but these are not a topic of this research. Therefore, we assume that the appropriate data is already displayed in the spreadsheet.

graphical element in the drawing window and then select cells, rows, and columns of data in the spreadsheet to specify that they are related.

To create a graph like figure Myers-Color-Plate-1-c, the user would first draw the axes approximately the right length, and then draw a single red rectangle on the horizontal axes, and a single blue rectangle overlapping in front of it. The link-boxes appear at each axis as the rectangles are drawn. The user selects the red rectangle (or its link-box) and then selects cell B2 in the spreadsheet to be associated with that rectangle. Because the rectangle was drawn at the horizontal axis, Gold assumes that the important graphical variable for the rectangle is the height, and therefore assumes that the value in the spreadsheet should be mapped into the height of the rectangle. However, it does not yet know the range of legal values. Next, the designer selects the blue rectangle and selects cell C2. Based on this, Gold assumes that the height of the red rectangles correspond to the column of values starting with B2, and that the height of the blue rectangles corresponds to the column of values starting from C2. To find the end of the column in the spreadsheet, Gold uses a heuristic where it searches down from the selected cell until it finds a cell with a different data type. If this guess is incorrect, the user can directly directly add or delete bars, or else edit the link-box associated with the axes, which shows the full range of values used. From the range of values in the spreadsheet, Gold determines the minimum and maximum value, sets up and labels the axes, and then draws rectangles for all of the values in the spreadsheet. Note that in the original example drawings, the user did not have to draw the rectangles at the correct heights based on the data; the heights are adjusted based on the actual data after the second example is provided.

Suppose, on the other hand, that the user wanted the color of the rectangles to visualize a different data value from the height, as in figure Myers-Color-Plate-2-c, where the color represents the country. Here, the user could again draw two rectangles, but the second one would be spaced differently (figure Myers-Color-Plate-2-a). The user would then associate cell B2 with the first bar as before, but B3 (instead of C2) with the second bar. Gold infers that the data series for the heights of the rectangles is column B, and that it does not know how to determine the color. Therefore, a color key is displayed (figure Myers-Color-Plate-2-b), and the user can specify column D2 in the first link-box. From this, Gold picks a third color different from the two the user supplied and creates the color key of figure Myers-Color-Plate-2-c. To specify a different color, the user would simply select the color key rectangle and change its color using the standard color palette. Note that after the user drew the first two bars in figure Myers-Color-Plate-2, an alternative interpretation might be the same as figure Myers-Color-Plate-1, with pairs of bars. This is not the default guess in figure Myers-Color-Plate-2 due to a heuristic that notices that the horizontal distance from the first bar to the second bar is about the same as the distance from the origin to the first bar.

If Gold guesses wrong about the mapping of data, or when users want to be more explicit about the mapping, then they can directly edit an element of the picture to be correct, and Gold uses this new information to refine its guesses. Alternatively, the link-boxes can be directly edited to specify the desired values.

HOW IT WORKS

Although there is an enormous variety of business chart styles, it turns out that they are constructed out of only a few primitives, with very standard composition rules. For example, bar charts, column charts, and stacked bar and column charts are all composed of rectangles that change in a single dimension. The problem is that the primitives can be combined in any fashion. Thus, providing all possible options is a combinatorial impossibility. By encoding the combination rules as heuristics, we can interpret the example drawings without needing to put all possible combinations in a menu.

An important issue is what properties of objects to support. Based on an analysis of the the types of charts provided, we have identified the following primitives and properties (where "color" is used to signify any filling style or hashing):

Graphical element	Parameters
bars	position, height, width, color
marks	position, size, color, shape (dots, squares, stars, etc.)
lines	position, color, line-width
pie segments	percentage of whole, color
labels	text string, position, color

Note that the primitives are "overloaded" since the same primitive object can be used for different purposes. For example, the bar object is used to draw horizontal bars, vertical columns, or floating rectangles in a scatter plot. It can even be used to draw a rectangle around a label to serve as a decoration. Lines can be used to attach data values in a line chart (the bottom of figure 1), or in vertical ranges (the top section of figure 1), or to draw an arrow that points to a special value (discussed below). Text strings can be uses as labels on axes, labels in the chart that display data (e.g., figure Myers-Color-Plate-3), markers for special values, captions, or decorations. By providing only a small number of primitives, we significantly reduce the number of concepts and terms the user needs to learn, and thus make the interface simpler. Heuristics are used to guess the role the graphic object is playing, and the link-boxes provide feedback to allow the user to monitor and edit the inferences.

In general, when the user draws two objects, Gold determines what is different between them, and assumes that all differences should be explained by different data. Thus, if rectangles are drawn with different colors, heights and horizontal positions, Gold expects to find three data values to cover these (but the user can explicitly map the same data value to multiple properties, to provide a redundant visualization).

Parsing the picture is made significantly easier because the

system has knowledge about the types of charts that are usually drawn. For example, it is most common for bars to be vertical with fixed widths, or horizontal with fixed heights. Bars used in pairs usually use the same colors for each pair. Gold uses five classes of heuristics to interpret the example picture: neatening, determining the type of chart, data typing, data mapping, and creating axes:

Neatening Heuristics - A few heuristics are used to tidy the chart, for example to align bars with the axes even if they are drawn a little off.

Chart Determining Heuristics - Based on the kinds of elements and their placement, Gold determines the type of chart for each component. Some examples:

- If a rectangle has one end on the axes, the system assumes that a value controls the other end to make a conventional bar chart.
- If a line, bar or mark has neither end near an axis, the system assumes that both ends (or the center and the size) must be accounted for (unless all the example marks are the same size). For example, two different data values control the top and bottom of the vertical lines for the temperature in the center of figure 1 (the daily maximum and minimum temperatures). As soon as the user selects the data to control one end, Gold will display objects of constant size for all data values, and allow the user to map the other end (or size) later.
- If two lines connect at an end-point, then the system assumes that the lines are connecting data points, as in the humidity graph at the bottom of figure 1. There might also be additional marks at the end points, as in figure 2.

Data Type Heuristics - These are used to determine types and properties of data in the spreadsheet.

- The inferred type of the data (nominal, ordinal, quantitative; dates, money, general numbers, names, etc.) is used to help guide the mappings and generalizations. These will be important, for example, to allow Gold to appropriately line up the values for graphs like figure 1, even though the temperature and humidity data is by day and the precipitation is by months. The knowledge about time will be adapted from the SAGE data display system [16].
- Other heuristics are used to find the range of data by looking in the spreadsheet to see where the data values change type. For example, in figure Myers-Color-Plate-1, after the user selects B2, Gold finds the range B2-8 since they are all numbers but B1 is a string and B9 is blank. If the guess is incorrect, the user can draw an additional object, delete an object, or else edit the appropriate link-box.

Data Mapping Heuristics - These heuristics determine how the values in the spreadsheet should be mapped to the graphical properties. For example:

- When objects are drawn with different colors or shapes, the system sees if these can map into different ranges of data. For example, the dots and stars in figure 2 map to different sets of data, as do the colors in figure Myers-Color-Plate-1. If not, Gold creates a key (as in figure Myers-Color-Plate-2) and allows the user to specify the mapping. This means that color and shape cannot currently be used to represent a continuous variable (but we plan to provide an explicit command to support this).

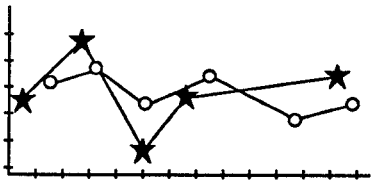


Figure 2:
The dots and stars correspond to different data series.

- If a string is placed inside the graph (not on an axes), the system checks if it is close to another object, for which it might serve as a label. If so, then the system looks into the spreadsheet data associated with the object to see if the value in a nearby cell is the same as the string (see figure Myers-Color-Plate-3).
- To determine how many examples are needed before the system can map the data, Gold looks at both the example graphics and the data. In many cases, the system can generalize from a single example. If the user has drawn a single graphical element (e.g., one bar) and if the spreadsheet is blank in one direction or if the user selects an entire row or column of the spreadsheet (rather than a single element), then Gold immediately uses the values to generate the rest of the objects for the chart. If there should be multiple bars (as in figure Myers-Color-Plate-1 and -2), then clearly the user will have to draw at least two bars.

Axis Creation Heuristics - These are used to determine the size and labeling of the axes:

- The system uses the size of the example axes, or if there are no axes, then the size of the current window, to be the size of the desired maximum range.
- To find labels for the axes, Gold looks above and to the left of the values from the spreadsheet for strings that might be labels. These are placed on the chart, as shown in figure Myers-Color-Plate-1-c. However, these text strings can be directly edited. Font and style changes are remembered, but if the user edits the text, then Gold will try to guess how the new string is composed by comparing it with values in the spreadsheet. If there are no exact matches, then the string is assumed to be a constant. The same heuristics will be applied if the user places an example string near an axis.
- Other heuristics will interpret short lines and numbers drawn near the axes as minor and major tick marks, and will interpolate or extrapolate from one or two examples.

As in demonstrational systems for other domains, such as graphical user interfaces [9, 14] and tables and text styles

[12], we use a domain-specific, knowledge-intensive rule-based system to implement the heuristics. As with all systems that use heuristics [15], Gold's rules were developed empirically. Because there are a small number of rules appropriate to any graphical object, sophisticated techniques from expert systems are not needed. For example, the rule-ordering has been adjusted experimentally to achieve the desired results, and we simply use the first rule that applies. The heuristics can create a wide variety of graphs with very little input from the user, and we will continually refine the heuristics based on frequent informal user trials.

FEEDBACK AND EDITING

Two of the key issues in systems using heuristics are the feedback to the user to show what the system is guessing, and the ability of the user to control and edit the resulting picture [15]. Gold provides a number of mechanisms to make these straightforward. The link-boxes show the cells of the spreadsheet that Gold has assigned to each axes, key, and graphical object. The user can directly edit the values in the link-boxes to specify or change the referent. To avoid clutter, the link-boxes are removed from the screen once they have been filled, but there are commands to bring back all or selected link-boxes.

As soon as the system generalizes from an example, it immediately draws new marks or bars that correspond to the rest of the values in the spreadsheet. If the system has generalized incorrectly, the user can simply edit one of the system-created objects so that it looks right, and Gold will use this additional information to refine its inferences. In this way, fixing incorrect inferences uses the same direct manipulation editing that the user is familiar with, rather than requiring extra mechanisms like question-answering [9], generated text [14], or special highlighting [2].

Similar editing techniques are used if the user wants to edit the display, possibly to explore the data using new views. The user can simply select portions of the display and draw replacements.[2] For example, to change the bars in figure Myers-Color-Plate-2 to dots, the user will be able to simply draw some dots, and use the replace command to replace them for the bars. To put time on the vertical axis instead of the horizontal one, the user will simply select and move the date labels to the left axis. Similar heuristic knowledge-based inferencing will be used to interpret the edits as for the initial drawings. Note that unlike the graphical replace in Chimera [6], here the *semantics* of the graphics will be used to determine how the replacement should be performed. For example, if bars of different colors are to be replaced by marks of different shapes, Gold will know how to perform the mapping, whereas Chimera could only replace the bars with a marks of different colors, but the same shape.

[2]Gold does not yet support changing the *data* through edits to the graphics, so all edits are assumed to be changes to the *display* of the data. Editing the data can currently only be performed in the spreadsheet. Eventually, we will probably provide a mode where the data can be edited through direct manipulation of the graphics.

SPECIAL ITEMS

Fancy charts such as those used by *USA Today* replace the bars and marks with arbitrary graphics. For example, different sized pictures of fish, people, or oil barrels might be used. Gold will make these kinds of charts easy to create by allowing any of the primitives to be replaced by an arbitrary picture which the user can draw in Gold or import from another drawing program. The picture will appear in the pop-up marks menu, and can then be placed and sized as easily as a star, diamond, or circle.

In many graphs, a particular item should be highlighted. For example, in figure 1, the highest and lowest temperatures are marked with a label and arrow. To achieve this in Gold, the user will draw an example of the special objects, or edit a property of an existing object (for example, to make a special bar of a graph be highlighted), and use a menu command to declare this to be a "marked" object. Gold will then try to identify why that item was chosen. For example, it might be the largest or smallest values (as in figure 1), the item with a particular value, or if nothing else applies, simply the specific item selected. A link-box will show Gold's inference. As a last resort, the user will be able to specify a spreadsheet cell which can contain a formula to compute the items to be highlighted. Unlike other charting programs which throw away these "decorations," Gold will recalculate which items to highlight when the data associated with the graph changes. If the marker is an extra object, Gold will check to see if it contains any values from the spreadsheet (as in figure 1) and adjust these as well based on the new data.

STATUS AND FUTURE WORK

The Gold prototype editor is being implemented using Garnet [11], and development is on-going. Gold currently supports many of the standard 2-D visualizations like those produced by Microsoft Excel and DeltaGraph. There are currently problems with the code for axes (which is why the ticks in the figures are wrong) and we cannot yet handle linked axes as in figure 1. We have also not yet performed user studies to refine the heuristics and verify the approach. After finishing the design described in this article, we plan to add map-based visualizations, where one or both coordinates are based on geographical data. This will allow Gold to support pictures like the famous Napolean's march [20, p. 41]. Hierarchical and network diagrams should also be easy to add, where the nodes and arcs can be arbitrary graphics which visualize various data values. For example, we envision that if the nodes should have a different color if they lie on a critical path, this could be directly demonstrated. 3-D visualizations, such as 3-D bar charts, will also be supported in the future. Many 3-D visualizations are based on straightforward projections of the primitives we already support. For example, a 3-D column chart uses parallelograms instead of rectangles for the bars. Contour charts are natural extensions to 3-D of map-based visualizations.

Algorithm animation and program animation are used to debug and investigate programs while they are running [10]. The graphics used in these systems are similar to some of the pictures that Gold can produce, so it would be

interesting to explore whether Gold can be used to design these visualizations. Then, techniques as in Marquise [14] could be used to add interactive behaviors to the graphics, so the user could edit the data through the pictures. The resulting graphic visualizations would then comprise a complete user interface for viewing and editing data, and therefore would be a complete direct manipulation, data-driven user interface created entirely by demonstration.

We also want to investigate generalizing the particular chart drawn by the user into a chart *style* which can be saved in a library and applied to different data. This might work in conjunction with a retrieval mechanism like SAGEBOOK [17].

CONCLUSIONS

The Gold data visualization editor allows users to directly draw examples of the desired pictures for conventional business graphics. This takes advantage of the skills users already have for drawing pictures in a drawing editor, while still eliminating the tedium of having to draw all points or to make the sizes exact. In addition, the generated displays change if the data changes. Using demonstrational techniques is particularly appropriate for the data visualization domain since the desired result is graphical and it is much more effective for the user to draw only a few example values. Furthermore, more kinds of charts can be specified than with a program like Excel or even DeltaGraph since arbitrary properties of objects can be mapped to data and users have more control over the display of the graphics, marks, axes and labels. For example, the other charting programs cannot create figure Myers-Color-Plate-3 where the text string represents data. Gold suggests that demonstrational techniques make the specification of the desired display much more direct, intuitive and effective for users, and it provides a good platform to explore many more exciting applications of demonstrational technology.

ACKNOWLEDGEMENTS

This research was partially funded by NSF grant number IRI-9020089.

REFERENCES

1. Stephen M. Casner. "A Task-Analytic Approach to the Automated Design of Graphic Presentations". *ACM Transactions on Graphics 10*, 2 (April 1991), 111-151.

2. A. Cypher. EAGER: Programming Repetitive Tasks by Example. Proceedings SIGCHI'91, N.O., LA, Apr, 1991, pp. 33-39.

3. Allen Cypher (Ed.) *Watch What I Do: Programming by Demonstration.* MIT Press, Camb., MA, 1993.

4. DeltaPoint, Inc. DeltaGraph V1.5. 200 Heritage Harbor, Suite G. Monterey, CA 93940.

5. S. Hudson and C. Hsi. A Synergistic Approach to Specifying Simple Number Independent Layouts by Example. Proceedings INTERCHI'93, Amsterdam, The Netherlands, Apr, 1993, pp. 285-292.

6. D. Kurlander and E.A. Bier. Graphical Search and Replace. Proceedings SIGGRAPH'88, Atlanta, GA, Aug., 1988, pp. 113-120.

7. Jock Mackinlay. "Automating the Design of Graphical Presentation of Relational Information". *ACM Transaction on Graphics 5*, 2 (April 1986), 110-141.

8. D.L. Maulsby and I.H. Witten. Inducing Procedures in a Direct-Manipulation Environment. Proceedings SIGCHI'89, Austin, TX, April, 1989, pp. 57-62.

9. Brad A. Myers. *Creating User Interfaces by Demonstration.* Academic Press, Boston, 1988.

10. Brad A. Myers. "Taxonomies of Visual Programming and Program Visualization". *Journal of Visual Languages and Computing 1*, 1 (March 1990), 97-123.

11. B.A. Myers, *et.al.* "Garnet: Comprehensive Support for Graphical, Highly-Interactive User Interfaces". *IEEE Computer 23*, 11 (Nov. 1990), 71-85.

12. B.A. Myers. Text Formatting by Demonstration. Proceedings SIGCHI'91, New Orleans, LA, April, 1991, pp. 251-256.

13. Brad A. Myers. "Demonstrational Interfaces: A Step Beyond Direct Manipulation". *IEEE Computer 25*, 8 (August 1992), 61-73.

14. B.A. Myers, R.G. McDaniel, and D.S. Kosbie. Marquise: Creating Complete User Interfaces by Demonstration. Proceedings INTERCHI'93, April, 1993, pp. 293-300.

15. B.A. Myers, R. Wolf, K. Potosnak, and C. Graham. Heuristics in Real User Interfaces. Proceedings INTERCHI'93, Amsterdam, The Netherlands, April, 1993, pp. 304-307.

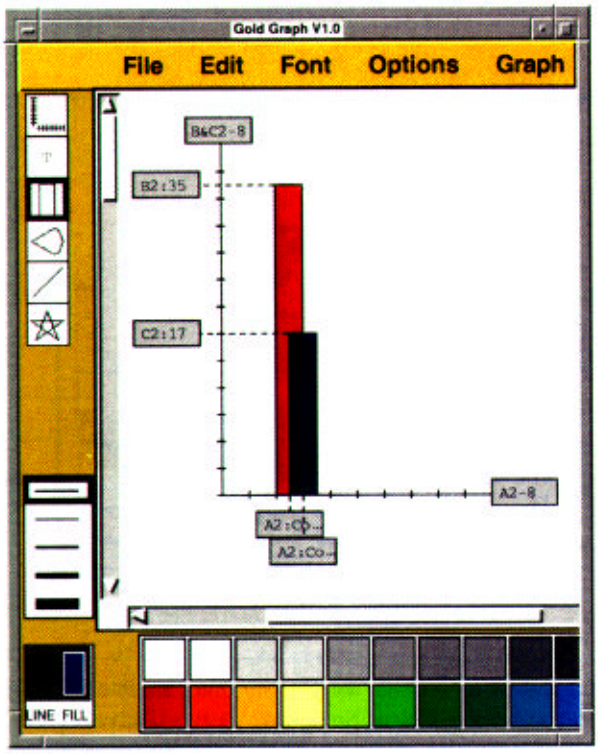
16. S.F. Roth and J. Matthis. Data Characterization for Intelligent Graphics Presentation. Proceedings SIGCHI'90, Seattle, WA, Apr, 1990, pp. 193-200.

17. S.F. Roth, J. Kolojejchick, J. Matthis, and J. Goldstein. Interactive Graphic Design Using Automatic Presentation Knowledge. Proceedings SIGCHI'94, Boston, MA, April, 1994.

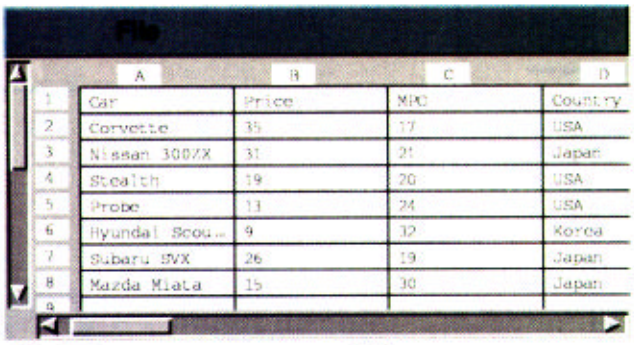
18. J.T. Stasko. Using Direct Manipulation to Build Algorithm Animations by Demonstration. Proceedings SIGCHI'91, N.O., LA, Apr, 1991, pp. 307-314.

19. J.T. Stasko, A. Badre, and C. Lewis. Do Algorithm Animations Assist Learning? An Empirical Study and Analysis. Proceedings INTERCHI'93, Amsterdam, The Netherlands, Apr, 1993, pp. 61-66.

20. Edward R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Conn., 1983.

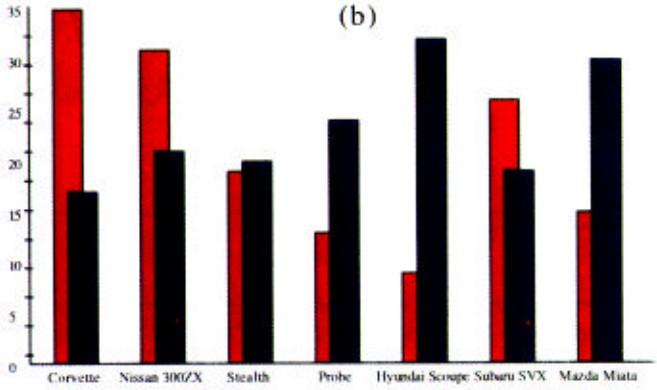


(a)



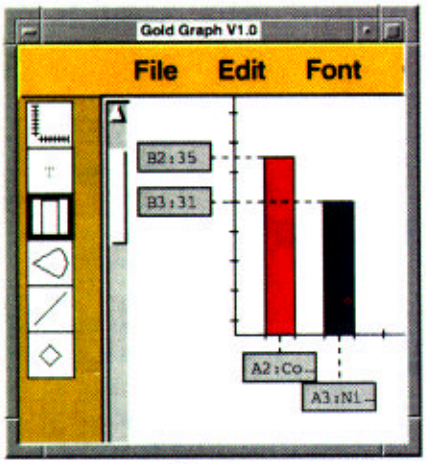
	A	B	C	D
1	Car	Price	MPG	Country
2	Corvette	35	17	USA
3	Nissan 300ZX	31	21	Japan
4	Stealth	19	20	USA
5	Probe	13	24	USA
6	Hyundai Scou...	9	32	Korea
7	Subaru SVX	26	19	Japan
8	Mazda Miata	15	30	Japan

(b)



(c)

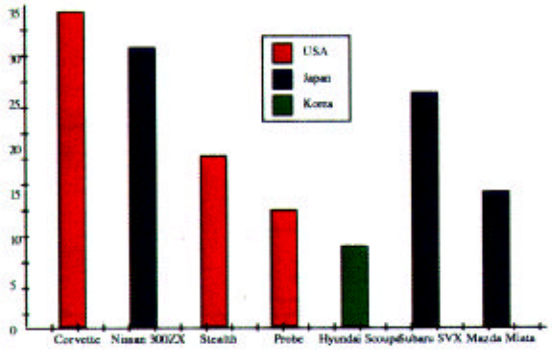
Myers, Color Plate 1: The Gold interface showing a paired column graph being constructed. The left edge of the window in (a) contains the palette of objects to be created, which are axes, text, rectangles (bars), pie-chart wedges, lines, and marks. Clicking on the marks brings up a pop-up menu of various graphics. The bottom and lower-left of the window contain the line and filling style palettes. The gray rectangles are "link-boxes" that show Gold's inferences of the relationship of the bars to the spreadsheet data, shown in (b). (c) shows the resulting graph using all of the data.



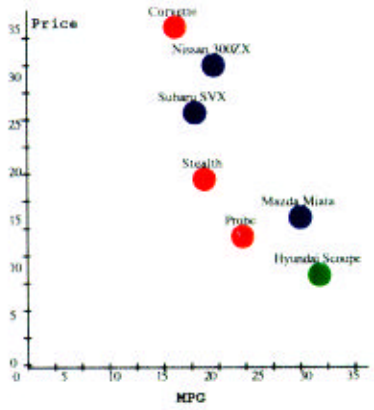
(a)

(b)

Myers, Color Plate 2: In this chart, the color of the bars corresponds to the country (see Myers Color Plate 1-b, column D) and the height corresponds to the price (column B). (a) shows what the user draws, causing Gold to create a color key (b). (c) shows the result.



(c)



Myers, Color Plate 3: The strings represent values from the spreadsheet. This kind of chart cannot be created using Excel or DeltaGraph.